(LA IMPOSIBILIDAD DE) EL COMPRESOR INFINITO

Pau Garcia i Quiles

Estudiante de Ingeniería de Telecomunicaciones e Ingeniería Informática
Universitat Politècnica de València

pgq@poboxes.com

Desde hace muchos años vemos anuncios de empresas que dicen haber inventado compresores de propiedades asombrosas: ratios de compresión 100 a 1, 1000 a 1, etc. ¿Por qué no los usamos si las ventajas son tan evidentes?

INTRODUCCIÓN

Bit, byte, compresión, tipo de datos... terminología que hace diez años hubiera dejado boquiabierto al más templado es hoy en día de uso corriente. Gracias al avance de la Sociedad de la Información, la informática y su vocabulario se han expandido como la pólvora, a la par que Internet se introducía en miles de hogares y empresas.

Además de hacer la declaración de la Renta y comprar libros, Internet ofrece muchas otras posibilidades. El deseo de transmitir o almacenar cada vez mayor volumen de datos sin cambiar el medio (línea telefónica, tarjeta de memoria, etc.) hace que se trabaje intensamente en el campo de la Teoría de la Información, especialmente, en la compresión de datos.

Un compresor es un algoritmo que disminuye el espacio ocupado por cierta información. Cada año vemos en los medios científicos y especializados (New Scientist, Nature, Byte, ZDNet, etc.) varios anuncios de empresas que aseguran haber superado el límite teórico (límite de Shannon) de compresión de datos. Generalmente, estos anuncios van asociados a peroratas como «necesitamos más dinero para terminar las investigaciones», «tenemos la tecnología pero hace falta un socio interesado en comercilizar la tecnología» (*ergo* gastar dinero en la empresa), etc. ¿Le suena de algo?

Un estudiante irlandés de 16 años consiguió engañar a un jurado de 12 personas y ganó el Premio al Joven Científico del Año 2002 con su «invento», un navegador de Internet llamado XWebs, que supuestamente multiplicaba por cuatro la velocidad de navegación. ZeoSync anunciaba en su página nada menos que cinco tecnologías relacionadas con la compresión de datos: codificador (BinaryAccelerator™), compresor (BitPerfect™), secuenciador (Zero Space Tuner™), transformador de dominio (Relational Differentiation Encoding™) y el conjunto de todo (TunerAccelerator™). Pegasus Technologies afirmaba que podía guardar 1.28 Gigabytes en un disquete de 3.5" (cuya capacidad normal es 1.44 Megabytes) gracias a su tecnología HyperDrive™ (de la que llegaron a obtener una patente –cosa fácil en USA, todo sea dicho–). Web Technologies nos asombraba con su compresor DataFiles/16™, que comprimía cualquier archivo mayor de 64 KBytes a un dieciseisavo de su tamaño. David C. James obtuvo una patente en USA para su tecnología HyperSpace™, que alardeaba de comprimir datos aleatorios. Premier Research Corporation hizo una aparición estelar con su compresor Minc, otro que decía poder comprimir datos aleatorios. Pixelon se aprovechó de la burbuja de las puntocom para vender humo y dilapidar 35 millones de dólares USA de sus inversores, mientras decía haber inventado un compresor de vídeo y audio de propiedades casi mágicas. Todas estas empresas han desaparecido.

Para entrar en materia, veamos un método sencillo de compresión. Si sustituímos las repeticiones de una letra por el número de veces que se repite podríamos comprimir esta secuencia:

AAAAAAAAAAAAAAABAAACDEEEEEECC
CCBBBFTTTTWCCAAAAAJJJLLLLL

como:

15AB3ACD6E4C3BF4TW2C5A3J5L

Lo que antes ocupaba 55 caracteres, ahora ocupa sólo 26: ¡una disminución del 53%!

En el ejemplo anterior hemos usado una versión bastante pedestre de un algoritmo de codificación por longitud de recorrido (Run-Length Encoding, RLE), pero que muestra la idea básica: eliminar aquello que se repite.

Los compresores eliminan la redundancia, posibilitando que una conexión a Internet lenta permita transmitir grandes volúmenes de datos, que se puedan realizar llamadas telefónicas y videoconferencias por Internet, etc. En resumen: permiten exprimir más nuestra capacidad de transmisión y almacenamiento de datos.

CONCEPTOS BÁSICOS DE TEORÍA DE LA INFORMACIÓN

La Teoría de la Información es la rama de la Matemática que estudia qué es la información, cómo medirla, cómo representarla y cómo transmitirla. Este campo es uno de los más jóvenes de la Matemática: su origen es el artículo *Una teoría matemática de la comunicación*, publicado en 1948 por Claude Shannon.

Antes que nada, ¿qué es la información?

La definición formal dice que la «información» de un mensaje es *la medida en que dicho mensaje despeja la incertidumbre sobre algo*. Con esta definción estamos suponiendo que la información se refiere a algo que desconocemos, es decir, algo que desde nuestro punto de vista tiene un componente aleatorio.

En Teoría de la Información se dice que los mensajes proporcionan la información, así que hablaremos indistintamente de «mensaje» o de «información», porque ésta última está asociada al primero. También se dice que los mensajes son emitidos por una *fuente de información*, que se supone aleatoria (así que no podemos predecir qué mensaje va a salir de la fuente). Debido a que la fuente es aleatoria, la Estadística nos permite caracterizarla usando una variable aleatoria *X*.

La fuente de información puede ser prácticamente cualquier cosa: una persona dictándonos los números que salen del «bombo» de la Lotería, alguien dictándonos nombres que lee al azar de la guía telefónica, un archivo de ordenador, etc. En todos estos casos, podemos reducir los mensajes a caracteres alfanuméricos (texto y números), que en una computadora se codifican como combinaciones de unos y ceros.

Nuestro alfabeto y nuestro sistema numérico tienen un conjunto finito de símbolos (las letras mayúsculas, las letras minúsculas, los números del 0 al 9, etc), pongamos, por ejemplo, que son 500 símbolos (en un caso genérico, hablaríamos de un alfabeto formado por N símbolos). Debido a esto, los valores de la variable aleatoria *X* (es decir, los mensajes) son también finitos y son exactamente los mismos (si los mensajes son de un símbolo, la fuente podrá emitir como mucho N=500 mensajes diferentes).

Matemáticamente, la información de un mensaje x_i (donde i es el número entre 1 y el número total de mensajes posibles N; en nuestro ejemplo, i varía entre 1 y 500) se caracteriza con un logaritmo en base 2 y la probabilidad de que la fuente X emita ese mensaje x_i:

$$I(x_i) = -\log_2 P(X = x_i)$$

Debido al signo negativo, cuanto más improbable es un mensaje, más información aporta. Por ejemplo, aporta mucha más información decir «hoy en el Ecuador hace frío» que decir «hoy en el Ecuador hace calor», porque lo primero es mucho más improbable que lo segundo.

Otro concepto de Teoría de Información que necesitaremos es la entropía. La idea es muy similar a la de la Termodinámica: la entropía mide lo desordenada que está la energía, en nuestro caso, lo desordenada que está la información. La función entropía da la cantidad media de información de la fuente de mensajes modelada por la variable *X*, es decir, la cantidad media de información que nos proporciona un mensaje x_i sobre *X*:

$$H(X) = \sum_{i=1}^{N} P(x_i) I(x_i)$$

Como consecuencia de esta definición, cuanto mayor es la entropía de una fuente, más información tiene cada mensaje, es decir, más improbable (difícil de predecir) es el mensaje.

Así pues, cuanto mayor es la entropía de una fuente, más difícil es comprimir los mensajes que emite.

TIPOS DE COMPRESORES: LOSSY Y LOSSLESS

En compresión de datos se distingue entre dos tipos de compresores, los compresores sin pérdida (*lossless*) y los compresores con pérdida (*lossy*).

La diferencia es evidente: cuando comprimimos un fichero de ordenador con un compresor lossless, al descomprimirlo obtenemos exactamente el mismo fichero que teníamos en origen, porque el compresor no produce pérdida de información. Con un compresor lossy, en cambio, es imposible recuperar el fichero tal y como era en origen, porque el compresor ha producido pérdida de información. Como contrapartida, los compresores lossy alcanzan niveles de compresión mucho mayores que los compresores lossless (a costa de la calidad del fichero obtenido).

Hay aplicaciones en las que es imprescindible usar compresores lossless, por ejemplo, en un documento de un procesador de texto: si usáramos un compresor lossy, perderíamos información, y al decomprimirlo, podríamos encontrarnos con que nos faltan frases, párrafos o imágenes.

En otras aplicaciones, en cambio, se puede usar sin problemas un compresor lossy. Los ejemplos más claros son la voz y la imagen. Como ni nuestro oído ni nuestro ojo son perfectos, una disminución moderada de la calidad es inapreciable. Los conocidos formatos MP3, MPEG, AVI, etc. son formatos de compresión lossy: se consigue que el audio o el vídeo ocupen poco espacio a costa de disminuir su calidad.

Después de ver en qué consiste la compresión lossy y la compresión lossless, podemos intuir que:

- La compresión lossless, al ser sin pérdida de información, tiene un límite máximo de compresión
- La compresión lossy, al ser con pérdida de información, no tiene un límite máximo de compresión: podemos comprimir tanto como queramos. Obviamente, cuanto más comprimamos, menos información tendremos (es decir, peor será la calidad final obtenida). En el caso límite, tenemos un 100% de compresión, o lo que es lo mismo, no tendremos nada de información.

Por supuesto, siempre que se anuncia uno de estos «compresores mágicos» se habla de compresores lossless, que son los que presentan mayores dificultades. Conseguir un compresor lossy con un nivel de compresión 1000 a 1 es fácil (eso sí, la calidad se resentirá).

En adelante, hablaremos sólo de compresión lossless, que es la que nos interesa en este artículo.

SUSTITUCIÓN DE CÓDIGOS

Las ideas básicas que debemos tener en mente son las siguientes:

- Queremos conservar toda la información
- La compresión lossless es la única que conserva toda la información
- Los compresores lossless actúan eliminando la redundancia

En la introducción ya vimos una forma de eliminar redundancia: los algoritmos RLE. Veamos ahora otra forma: los algoritmos de sustitución de códigos (normalmente se usan los dos conjuntamente: RLE y sustitución de códigos).

La sustitución de códigos para comprimir una cadena C (formada por varios mensajes enlazados uno a continuación de otro) consiste en cambiar la forma en que se representa esa cadena. A la hora de decidir cómo se hará el cambio, lo fundamental es la probabilidad con que se presenta parte de la cadena (cada mensaje).

Veamoslo con un ejemplo. Imaginemos una fuente que emite 5 mensajes: M1, M2, M3, M4 y M5, con probabilidades 35%, 25%, 20%, 12% y 8% y longitudes (en caracteres) 10, 7, 6, 4 y 3
Si dejamos que la fuente emita 1000 mensajes, teniendo en cuenta las probabilidades y las longitudes de los mensajes, la cadena emitida ocupará 7170 caracteres

Msj	Representación	Prob.
M1	1 1 1 1 1 1 1 1 1 1	35%
M2	2 2 2 2 2 2 2	25%
M3	3 3 3 3 3 3	20%
M4	4 4 4 4	12%
M5	5 5 5	8%

$$\frac{35x1000x10}{100}+\frac{25x1000x7}{100}+\frac{20x1000x6}{100}+\frac{12x1000x4}{100}+\frac{8x1000x3}{100}=7170$$

```
11111111111111111112222222444455533
33311111111111111111112222222222222
11111111111111111113333331111111113
33333222222211111111111111111155522
22223333331111111114444222222233333
31111111112222222222222244444444333
33222222222222233333311111111155511
11111115555111111111111111111333333
22222224444111111111....
```

Como el mensaje M1 (representado como 1111111111) aparece mucho (35% de las veces) y es muy largo (10 caracteres) y el mensaje M5 (representado como 555) aparece poco (8% de las veces) y es corto (3 caracteres) una forma sencilla de comprimir consiste en intercambiar las representaciones de los mensajes:

Msj	Representación	Prob.
M1	5 5 5	35%
M2	2 2 2 2 2 2 2	25%
M3	3 3 3 3 3 3	20%
M4	4 4 4 4	12%
M5	1 1 1 1 1 1 1 1 1 1	8%

Al hacer esta «compresión», la cadena de 1000 mensajes tiene una longitud de 5280 caracteres (hemos conseguido más de un 26% de compresión):

```
55555522222224444111111111333333555
55222222222222225555553333335553333
22222255555511111111112222222333335
55444422222223333335552222222222224
44444443333332222222222223333335551
11111111555111111111555555333333222
22224444555....
```

(ahora los 555 representan a M1, no a M5)

Podemos hacer lo mismo otra vez, intercambiando las representaciones de los mensajes M2 y M4:

Msj	Representación	Prob.
M1	5 5 5	35%
M2	4 4 4 4	25%
M3	3 3 3 3 3 3	20%
M4	2 2 2 2 2 2 2	12%
M5	1 1 1 1 1 1 1 1 1 1	8%

Tras esta nueva vuelta de tuerca, la cadena de 1000 mensajes ocupa sólo 4890 caracteres (hemos conseguido un 32% de compresión respecto a la cadena original).

El código más conocido, el Morse, es un código de este tipo. La letra más frecuente, la *e*, se representa por un punto (es decir, la letra más frecuente es la más corta). Por desgracia para el español, la *e* es la más frecuente en inglés, mientras que la más frecuente en español es la *a*. Algo similar ocurre con la *W*, codificada como .— (es decir, relativamente corta, lo que tiene sentido en inglés, pero ningún sentido en español).

Ahora que ya sabemos «grosso modo» cómo funciona la compresión por sustitución de códigos, vemos que unos mensajes disminuyen el espacio que ocupan (M1 y M2), otros lo aumentan (M4 y M5) y algunos no lo varían (M3). Como los mensajes que ocupan más aparecen poco, y los que ocupan menos aparecen mucho, el total del espacio ocupado es menor tras la compresión.

Pero, ¿qué ocurre cuando ya no se pueden hacer más cambios de representación? Dicho de otro modo, ¿qué ocurre si la información es completamente aleatoria (todos los mensajes tienen exactamente la misma probabilidad de aparición)? En este caso, no se puede comprimir la cadena, porque al hacer nuestros «intercambios de representaciones» no ganamos nada.

La conclusión obvia es que llega un momento en que se ha eliminado toda la redundancia (utilizando RLE) y ya no se pueden hacer intercambios de representaciones, así que no se puede comprimir más. Hemos llegado al «límite de compresión», que viene dado por la entropía de la cadena (o fichero) que deseamos comprimir, y cualquier intento de «recomprimir» la información ya comprimida será infructuoso.

Veamos un ejemplo sencillo de límite de compresión, con los mismos mensajes que antes, pero cambiando la probabilidad de aparición: ahora todos los mensajes son equiprobables (aparecen las mismas veces):

Msj	Representación	Prob.
M1	1 1 1 1 1 1 1 1 1 1	20%
M2	2 2 2 2 2 2 2	20%
M3	3 3 3 3 3 3	20%
M4	4 4 4 4	20%
M5	5 5 5	20%

Una cadena formada por 1000 mensajes tiene ahora 6000 caracteres. Intercambiemos las representaciones de M1 y M5:

Msj	Representación	Prob.
M1	5 5 5	20%
M2	2 2 2 2 2 2 2	20%
M3	3 3 3 3 3 3	20%
M4	4 4 4 4	20%
M5	1 1 1 1 1 1 1 1 1 1	20%

La cadena formada por 1000 mensajes vuelve a tener exactamente 6000 caracteres. No hemos conseguido nada. El lector puede comprobar que tampoco se gana nada

Antes de RLE		Después de RLE
555	?	35
4444	?	44
333333	?	63
2222222	?	72
1111111111	?	101

intercambiando las representaciones de M2 y M4, o haciendo cualquier otro cambio de representación. Lo único que quedaría por hacer es aplicar un algoritmo RLE para conseguir algo parecido a esto:

TRANSFORMACIONES

Las Matemáticas y la Física suelen recurrir a cambios de dominio (por ejemplo, pasar de dominio tiempo a dominio frecuencia) para facilitar determinadas operaciones.

Podríamos preguntarnos si haciendo algún procesado, tal como una Transformada de Fourier, una Transformada de Haar, o cualquier otro procesado de señal) a la cadena sin comprimir (preprocesado) o a la cadena comprimida (postprocesado) sería posible alcanzar una mayor compresión. Es decir, ¿cambia el límite de compresión al cambiar el dominio en que representamos la información?

La respuesta es «no». La razón es que la Teoría de la Información se basa en la probabilidad, y el modelo probabilístico ya incluye todas las posibles transformaciones que se puedan hacer, porque al cambiar

de dominio, lo único que cambiamos es el modelo probabilístico que usamos. Veamos la demostración formal (no es necesario entender la demostración para seguir leyendo el artículo; puede pasarla por alto si lo desea). Dado un modelo probabilístico P y una función reversible M, siempre existe un modelo probabilístico Q tal que Q(M(C)) = P(C), para cuaquier cadena C.

Como M es reversible, para cualesquier cadenas C y D (tales que C '« D), se cumple que M(C) '« M(D).

Podemos definir Q(C) tal que para cualquier cadena C, Q(M(C)) = P(C), es decir, dos modelos probabilísticos diferentes (uno en cada dominio) nos dan el mismo resultado. Así pues, el cambio de dominio no mejora la entropía de la fuente y por tanto, no mejora la compresión.

CONCLUSIÓN

Como hemos visto a lo largo del artículo, llega un momento en que seguir aplicando algoritmos de compresión (RLE, sustitución de códigos, etc.) no disminuye el tamaño final, sino que lo aumenta: hemos llegado al límite de compresión. Esto se puede ver incluso de forma intuitiva: si siempre se pudiese seguir comprimiendo, llegaría un momento en que cualquier cadena original se reduciría a un único

```
AAAA          → DF     → A
BBBBBBBBBB    → HRFD   → B
CCCCCCC       → CFD    → D
ADVCEFDD      → CDF    → D
AGHBCED       → AS     → A
ACVWREYIPGDD  → WECB   → B
```

```
A ─┬─ ¿ DF → AAAA ?
   └─ ¿ AS → AGHBCED ?

B ─┬─ ¿ HRFD → BBBBBBBBBB ?
   └─ ¿ WECB → ACVWREYIPGDD ?
```

carácter. ¿Cómo saber a cuál de las infinitas cadenas originales corresponde ése carácter? (es decir, a qué se tiene que descomprimir ése carácter)

Aunque las tecnologías de compresión usadas hoy en día no siempre son óptimas, cualquier anuncio de tecnología de compresión con límites de compresión muy superiores a los actuales es muy probablemente falso.

En cualquier caso, si una tecnología de compresión lossless que afirma superar el límite de Shannon, no hay duda de que mentirá, así que no vale la pena perder tiempo en ella. Ahora ya podemos contestar a la pregunta que se planteaba al inicio del artículo: no usamos esos compresores fantásticos porque no existen (ni existirán nunca: la analogía más clara sería una máquina de movimiento perpetuo).

Aún así, el lector osado puede aceptar el reto de Mike Goldman: 5000 dólares USA para quien consiga comprimir datos aleatorios, desafiando así el Teorema de la Complejidad de Kolmogorov (si se intenta comprimir un fichero de datos aleatorios, el tamaño del fichero comprimido más el tamaño del compresor siempre será mayor que el tamaño del fichero sin comprimir).

REFERENCIAS BIBLIOGRÁFICAS

[1] «Comunicación de datos»
J. R. Vidal Català, J. Martínez Zaldívar
Editorial de la Universidad Politécnica de Valencia, 1995

[2] comp.compression FAQ
Jean-Luc Gailly
http://www.faqs.org/faqs/compression-faq/

[3] There's no magic
Charles Bloom
http://www.cbloom.com/news/nomagic.html

[4] Information content and compression limit FAQ
Glyph
http://www.geocities.com/CollegePark/9315/infofaq.htm

ZeoSync, BinaryAccelerator, BitPerfect, Zero Space Tuner, Relational Differentiation Encoding y TunerAccelerator son marcas registradas de ZeoSync Corporation

Web Technologies y DataFiles/16 son marcas registradas de Web Technologies

Premier Research Corporation y Minc son marcas registradas de Premier Research Corporation.

Pegasus Technologies y HyperDrive son marcas registradas de Pegasus Technologies.

HyperSpace es una marca registrada de David C. James.

Pixelon es una marca registrada de Pixelon Corporation.

Agradecimientos especiales a Félix Ares y Jean René Moreau por sus valiosos consejos y correcciones.

AUTOR

***Pau Garcia i Quiles**, nacido en Elche (Alicante) en 1979. Ingeniero Técnico de Telecomunicación, especialidad de Telemática, por la Universidad Politécnica de Valencia. Actualmente haciendo en la UPV 2° de Ingeniería Informática y el Proyecto Final de Carrera de Ingeniería Superior de Telecomunicación ("Evaluación del algoritmo Rijndael para la generación de números pseudoaleatorios"). Intereses principales: Unix, seguridad informática y divulgación científica (dentro de la cuál se enmarca este artículo).*